DISTRIBUTED ASSIGNMENT OF CODES FOR MULTIHOP PACKET-RADIO NETWORKS

J.J. Garcia-Luna-Aceves and Jyoti Raju
Computer Engineering Department
School of Engineering
University of California
Santa Cruz, CA 95064
jj, jyoti@cse.ucsc.edu

Abstract— **This paper describes and analyzes a distributed algorithm for assigning codes in a dynamic, multihop wireless radio network. The algorithm does not require any form of synchronization and is completely distributed. The algorithm can be used for both the transmitter oriented and receiver oriented code assignment. The algorithm is proven to be correct and its complexity is analyzed. The implementation of the code assignment algorithm as part of the medium access control (MAC) and routing protocols of a multihop packet-radio network is discussed.**

I. INTRODUCTION

Using code division multiple access (*CDMA*) in packet-radio networks permits multiple stations within range of the same receivers to transmit concurrently, without interference. Several multiaccess protocols have been proposed and commercial systems have been deployed that take advantage of CDMA [1]. An important design consideration in a multihop packet-radio network using CDMA is the assignment of transmission codes to network nodes. In a large network, the number of transmission codes is smaller than the number of nodes, and senders and receivers must agree on which transmission code to use in a way that avoids interference as much as possible.

Interference in a CDMA network can be of two kinds. *Direct* interference consists of two neighboring nodes transmitting to each other at the same time. This can be avoided by using a random access protocol that schedules transmissions to the neighbor nodes in time to avoid collisions (e.g., FAMA [8]). *Secondary* interference consists of two senders transmitting to receivers in a way that the senders' transmissions interfere at at least one receiver. This type of interference can occur in two scenarios.Two stations unaware of each other's existence can transmit to the same receiver at the same time; this gives rise to the transmitter-oriented code assignment problem shown in Fig. 1. Here, A and C, which are two hops away from each other need to have different codes. The second case of secondary interference occurs when a station is transmitting to its neighbor and a third station's transmission to some station other than the stations involved in the first transmission causes an interference with the first transmission. This leads to the receiver-oriented code assignment problem illustrated in Fig. 2. Here, C is transmitting to D but causes interference at B, because B and D which are two hops away share the same code. As can be seen from the earlier two examples, the code assignment problem requires that no set of stations which are two hops away have the same code.

Several approaches have been proposed in the past for channel/code assignments. A tree-based protocol for broadcasting is discussed by Chlamtac and Kutten [4]; the distributed version of the protocol uses a travelling token, which incurs more overhead traffic on the network. Chlamtac and Kutten also show that designing a protocol such that the least amount of channel bandwidth is used for broadcast is NP-complete. A different approach to the token-based scheme consists of using control segments during which the schedule for the transmission segments is decided [5]. The transmitter-oriented code assignment is introduced in [11]. Here, quasiorthogonal codes are assigned to transmitters in a packet-radio network in such a way that hidden terminal interference is eliminated. There are many approaches based on the fact that the node needs to know the information of nodes two-hops away [6], [3], [9].

This paper presents a distributed code assignment algorithm. This algorithm assigns a code to each node in a way that no interference occurs after the algorithm converges, provided that the number of codes available for assignment is at least $d(d-1)+2$, where d is the maximum number of one-hop neighbors any node can have. This is because the algorithm assigns a transmission code to a node that is different than the codes assigned to nodes two hops away from the node. The algorithm is designed to be part of the MAC and routing protocols of a multihop packet-radio network. It is based on the asynchronous exchange of control messages that are part of the regular MAC and routing messages, and the information generated by any one node propagates up to two hops away from the node.

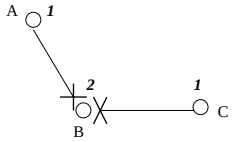

Fig. 1. Interference when two transmitters two-hops away have the same code

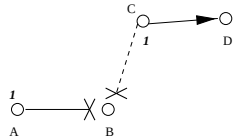

Fig. 2. Interference when two receivers two-hops away have the same code

Section II describes the algorithm in detail. Section III shows that the algorithm guarantees correct code assignments after convergence, provided that enough codes are available. Section IV addresses the algorithm's complexity.

This work was supported in part by the Defense Advanced Research Projects Agency (DARPA) under grant DAAB07-95-C-D157.

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
1997		**00-00-1997 to 00-00-1997**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
Distributed Assignment of Codes for Multihop Packet-Radio Networks	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **University of California at Santa Cruz,Department of Computer Engineering,Santa Cruz,CA,95064**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**		**5**	

II. The Algorithm

A. Network Model and Protocol Assumptions

To describe our code assignment algorithm, we model a multihop packet-radio network as an undirected graph $G = (V, E)$ where V is the set of nodes and E is the set of edges. Each node consists of a transceiver and a router. A link between two nodes i and j in G means that i can hear j's transmission and j can hear i's transmission. Each node uses an omnidirectional antenna for transmission and the network works in half-duplex mode, which means that a node cannot transmit and receive at the same time. A routing protocol is assumed to create and update the routing table used at each node. This routing protocol provides information about who the node's active neighbors are; this involves adding to the neighbor list new neighbors when they come up and deleting neighbors which are no longer active. The routing protocol is assumed to have some form of neighbor discovery mechanism such as a HELLO exchange [12]. Nodes process messages they receive and links transmit packets in the FIFO order.

We also assume a MAC protocol that allows stations to schedule their transmissions to the same receiver in a way that collisions of data packets to the receivers are avoided. The floor acquisition multiple access (FAMA) family of protocols [8] is an example of such a MAC protocol. According to FAMA, a sender transmits a request-to-send (RTS) to the receiver, which in turn transmits a clear-to-send (CTS) if it obtains the RTS free of errors. The RTS lasts longer than the propagation delay and the CTS must be larger than size of RTS + maximum round trip propagation delay + transmit-to-receive turnaround time, so that collisions due to hidden terminals are eliminated in the absence of erasures due to drastic node mobility. When the network is first brought up, all transmissions take place over a common signaling code using FAMA. As the stations decide on their codes, the stations use different codes for data transmissions.

A link is assumed to exist between two nodes only if there is good radio connectivity and the update messages of the routing protocol can be sent reliably. Node failures are modelled as all links incident on the node failing at the same time. A moving node is considered attached to all nodes with which the node can exchange messages with a certain probability of success; a moving node becomes detached from nodes with which it cannot exchange messages with this probability of success. The code assignment algorithm runs in conjunction with the routing protocol, which in turn runs on top of the MAC protocol.

B. Principles of Operation

The messages of the code-assignment protocol are called the *Code Assignment Messages (CAM)*. These messages are sent as part of the messages exchanged in the network's routing protocol. These messages could be lost due to changes in radio connectivity. Reliable transmissions of *CAM*'s is done with the help of retransmissions. After receiving a *CAM*, the receiver is required to acknowledge it to the node which sent it, by sending an explicit *ACK* message indicating that the *CAM* has been received and processed. Since, the network layer assumes the link to be of a broadcast nature, a node can send a message once and assume to have sent it to all the neighbors; however, each neighbor needs to *ACK* the message individually.

C. Information at Each Node

The following structures are needed for code assignment at each node.

- Priority List: Each node has a unique priority number assigned to it. This priority number allows two-hop neighbors with the same code to decide which one of them should keep the code and which must look for new codes. We decided to use the address of a node as its priority number, because addresses are unique.
  The priority list contains the priority numbers of the node, its one-hop and two-hop neighbors sorted in increasing order. Each entry in the list has a flag which is set to 1 if the neighbor is one-hop or 2 if the neighbor is two-hop. The codes assigned to the node and its neighbors are also listed in here.
- Neighbor List : This is a list of all the node's immediate neighbors, i.e. all its one-hop neighbors.
- Code Assignment Message Retransmission List (*CAMRL*): This list has one or more retransmission entries, where an entry is of the following nature.
  - The sequence number of the *CAM*.
  - A retransmission counter that is decremented every time the node sends a *CAM*.
  - An *ACK*-required flag that is actually a field of the size of the neighbor list. The bit corresponding to a neighbor is set if the *CAM* is yet to be acknowledged by the neighbor.

  The CAMRL permits a node to know which *CAM* is not acknowledged by some of the neighbors and needs to be sent again. The node retransmits a *CAM* when its retransmission entry in the *CAMRL* reaches zero. The retransmission counter of a new entry in the *CAMRL* is set equal to a small number (e.g., 3 or 4).
- Unassigned Code List (*UCL*): This list contains all the codes which are available, i.e. they are not being used by any of the node's two-hop neighbors.

D. Information Exchanged among Nodes

The *CAM*'s propagate only from a node to its neighbors and no further. Each *CAM* contains

- The address of the node which is sending the *CAM* along with its code.
- The addresses of the node's one-hop neighbors along with their codes.
- *ACK*'s to earlier *CAM*'s. An *ACK* entry specifies the source and the sequence number of the *CAM* being acknowledged.
- A response list of zero or more nodes which need to send an *ACK* for this *CAM*.
  If a single *CAM* is not large enough to hold all the codes and addresses, the information can be split up into more than one *CAM*.

CAM's are sent in the following situations.

1. A new node comes up. Its priority list only consists of its own address and its own code. It broadcasts this to all its neighbors.
2. When a node i detects a change of code by any of its one-hop neighbors, i makes the required changes in its priority list and sends a *CAM* to all its one-hop neighbors including the one that changed the code.

3. When a node i finds that a certain one-hop neighbor j is no longer active, it drops j from the priority list. This information is then conveyed to all its one-hop neighbors by a *CAM* which reflects the changes.

E. Updating the Priority List

A node updates its priority list either after detecting a change of code in one of its one-hop neighbors or after receiving a *CAM* which contains information about a change of code by its two-hop neighbors.

When a node notices a change of code in any of its one-hop neighbors, it makes a change in its own priority list and sends a *CAM* with its own address and the address of all its one-hop neighbors along with their respective codes. This is sent to all its one-hop neighbors. If a two-hop neighbor has changed a code then any of the following three situations might arise.

1. If the new code of the two-hop neighbor is not the same as i's code, then the new code in entered into the priority list in the entry corresponding to the two-hop neighbor.
2. If the new code of the two-hop neighbor is the same as i's code and the address of the two-hop neighbor is lesser than i's code, then i picks up a new code. The new codes of the two-hop neighbor and i are entered into the priority list.
3. If the new code of the two-hop neighbor is the same as i's code and the address of the two-hop neighbor is greater than i's code, then i retains its code and there is a temporary conflict till the two-hop neighbor changes its code. The two-hop neighbor will get to know of i's code because of the *CAM* sent by the one-hop neighbor of i. This conflict does not stop the data transfer in the network. It merely increases collisions.

In the first two cases the *UCL* has to be updated. All the codes used by the two-hop neighbors are marked as unavailable in the *UCL*.

F. Sending New and Retransmitted CAM's

Whenever a node i sends a new *CAM*, it must do the following steps

1. Decrement the retransmission counter of all the new entries in the *CAMRL*.
2. Delete the entries which correspond to entries in the new *CAM*.
3. Add an entry in the *CAMRL* for the new *CAM*.

If a certain *CAM* in the *CAMRL* has all its entries covered be a new *CAM* transmission, the old *CAM* is deleted from the *CAMRL*.

When the retransmission counter for a *CAM* retransmission entry expires, node i sends a new *CAM* which has the same data as the old *CAM*. However, it has a new sequence number and a new response list. This new response list specifies the neighbors which did not acknowledge the *CAM* earlier. The old entry in the *CAMRL* is deleted and a new entry created for the new retransmission.

Using the above retransmission strategy, a node can keep sending *CAM*'s till all its one-hop neighbors acknowledge it. However, after some time if a node gets no response from a neighbor, it considers the neighbor dead and does not send it any *CAM*'s and resets bits which correspond to this neighbor in the response lists of all the entries in the $CAMRL$.

G. Processing an ACK

The *ACK* entry bears the sequence number of the *CAM* it acknowledges. As soon as an *ACK* is received by a node, the node searches its *CAMRL* to find the entry with the required sequence number. If a match is found the node resets the *ACK* required flag for the neighbor which sent the *ACK*.

A node may receive an *ACK* for an entry which has been deleted due to more recent *CAM*'s to the same neighbor or for an entry which has undergone a change in sequence number due to more recent retransmissions. In that case, the node simply ignores the *ACK*.

H. Embedding Code Assignment Algorithm in MAC and Routing Protocols

The proposed code assignment algorithm can be used as an integral part of the MAC and routing protocol of a packet-radio network.

In our model, all stations have a common signaling code on which they implement an *RTS-CTS* exchange similar to the 802.11 protocol [10]. The data transfer can be done on the receiver's code for *ROCA* or on the transmitter's code for *TOCA*. The advantage of some form of code assignment is that a node's one-hop neighbor can receive a data transmission at the same time as the node.

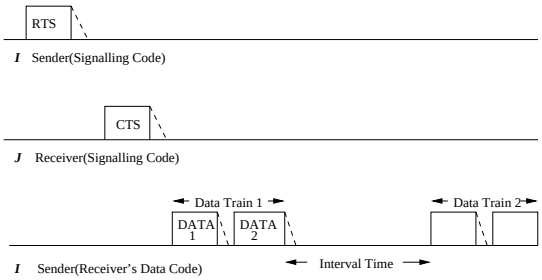

Fig. 3. Handshake and Data Transfer using Receiver's Code

We investigate the case for the receiver oriented approach (Fig. 3). Consider two nodes I and J which are immediate neighbors of each other. The node I needs to transmit data to J. I sends an *RTS* to J in which it specifies the code it expects J to have. It also specifies the number of data packets it plans to send. J send back a *CTS* which contains its code and the maximum number of data packets it can allow. After successfully receiving the *CTS*, the node I transmits its data using the code.

After the specified number of data packets have been transmitted and received, I and J switch back to the signaling code. If the *RTS-CTS* exchange also specifies the interval time between two packet bursts, then after an initial handshake I can switch to J's data code automatically without an intervening *RTS-CTS* handshake. The last packet of a data burst can contain control data which specifies if there are more data bursts following. If node J gets any requests which would require it to receive data from any node other than I during the time it has reserved for I, it will refuse the requests. Also, any *RTS* arriving(on the signaling code) at J while it is receiving data from I (on J's data code) would not destroy the data. This takes care of the multihop problem.

Now, with the availability of assured time for data transmission, a certain rate for data transmission can be guaranteed. This is use-

ful for real-time data traffic which requires constant delay. However, one has to keep in mind that if the topology gets constantly rearranged, such a rate cannot be sustained.

A similar treatment can be done for the transmitter oriented approach. Here, the *RTS-CTS* will exchange the sender's code.

CAM's need to contain information about a node's one-hop neighbors and their codes. If the distance metric in a routing algorithm is hop-count, then reading a routing update message from a node is enough for the code assignment algorithm to deduce a node's one-hop neighbors. However, the codes assigned to the neighbors are not present in the routing updates. With the addition of this field the routing updates can be used as *CAM*'s.

III. Correctness of the Algorithm

This section shows that the code-assignment protocol is correct under the assumption that an underlying protocol assures the following conditions.

1. A node detects within a finite time the existence of a new neighbor or the loss of connectivity with a neighbor. A node also detects within a finite time a change in the code used by a neighbor.
2. All messages transmitted over a radio link are received correctly and in the proper sequence within a finite time.
3. All messages and code changes are processed one at a time within a finite time and in the order in which they are detected.

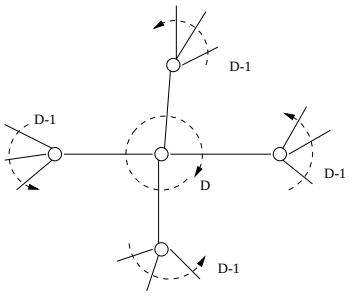

Fig. 4. The connectivity of a node in a network a maximum degree d

Another assumption we make is that we have a minimum of $d(d-1)+2$ codes, where d is the maximum degree of the network. One code is required for signaling and the use of the rest of them can be explained using Fig. 4. A node can have a maximum of d neighbors. Each of these neighbors can in turn have d neighbors. Consider the central node i. It has d one-hop neighbors and $d(d-1)$ two-hop neighbors. The node should not have a code in common with any of its two-hop neighbors. Thus, it is clear that we require $d(d-1)+1$ codes. This is a sufficient condition to obtain valid code assignments [9]. The network can still operate with fewer than $d(d-1)+2$, even though some two-hop neighbors will have the same codes. However, for the proof, we assume that $d(d-1)+2$ codes exist.

Additional assumptions are that there are a finite number of code changes upto time t_0, and that no more changes occur after that time, and that nodes can correctly determine which *CAM*'s are more recent than the others.

Correctness for this algorithm means that within a finite time after t_0, all nodes obtain information about the codes of their one-hop and two-hop neighbors. This allows them to calculate a code for themselves which is different from the codes of all their two-hop neighbors.

Definition 1 : There are two different ways a node can change its code.

1. The node can change its code to any of the unused two-hop codes(this includes codes of one-hop neighbors).
2. The node can change to a code that is already being used by a two-hop neighbor with an address greater than itself.

Definition 2: A node is said to have consistent information in its priority table if it has the most recent information about the change of codes of its one-hop and two-hop neighbors.

Theorem 1: *A finite time after t_0, all nodes have consistent information in their priority tables and the codes they decide based on this information are correct.*

To prove the above statement we need to show that the following conditions are satisfied:

1. All nodes eventually stop updating their priority list and stop sending update messages to their neighbors.
2. All nodes must have consistent code information in their topology databases within a finite amount of time after t_0.
3. If the information in the node is consistent and most recent, a node only makes a valid change of code and there are no deadlocks.
4. If all nodes make only valid change of codes, then we have a correct code assignment, such that no pair of two-hop neighbors have the same code.

Lemma 1: *All nodes eventually stop updating their priority list and stop sending update messages to their neighbors.*

Proof: First, note that there are a finite number of nodes in the network, and that by assumption a finite number of code changes can occur upto time t_0, after which no more changes in codes occur. Also by assumption, a change of code by a certain node is detected within a finite amount of time by the neighboring nodes. These neighboring nodes, in turn update their priority lists and send out at most one CAM to each of their neighbors. Therefore, for any change of code, there can be at most $d(d-1)$ messages sent.

A node which does not allow the protocol to terminate must be generating an infinite amount of CAM's. This is only possible if one of its neighbors is changing its code an infinite number of times. Since, we assume that there are no changes after t_0, this is not possible. Therefore, the protocol produces only a finite number of CAM's and the message transfer must stop within a finite time after t_0.

Lemma 2: *All nodes must have consistent code information in their topology databases within a finite amount of time after t_0.*

Proof: Consistent code information means that the node knows about all the recent code changes in its one-hop and two-hop neighbors. Consider some node i. A lower level protocol guarantees that i will detect a change of code in any of its one-hop neighbors within a finite time.

Consider the case of a node i whose two-hop neighbor k changes its code. This two hop neighbor is the one-hop neighbor of one of i's one-hop neighbors(j). We are guaranteed that j detects a change in k's code in a finite time. This detection causes a change in j's priority list, which in turn causes j to send a *CAM*

to all its one-hop neighbors which include i and k. Since, we assume that node j processes CAM's in a finite time, we always have the latest code change information at a node in a finite time after the last change, i.e. after t_0.

Lemma3: *If the information in the node is consistent and most recent, a node only makes a valid change of code and there are no deadlocks.*

Proof: When a node gets a *CAM*, it makes changes in its priority table and checks to see if any of its two-hop neighbors use the same code as it does. If no two-hop neighbor has the same code, the node does nothing. If a two-hop neighbor has the same code, then the node checks the address of the two-hop neighbor. If the two-hop neighbor has a larger address, then the node keeps its own code. If the two-hop neighbor with the same code has an address smaller than the node, then the node has to pick up a new code from the UCL. If there are $d(d-1)+2$ codes, there will always be some code available to pick. Thus, there is always a valid change of code. It can be seen that there are no possible situations where a node has to wait for any event to occur and therefore, there exist no deadlocks.

Lemma 4: *If all nodes make only valid changes of codes, then we have a correct code assignment in which no two-hop neighbors have the same code.*

Proof: Consider a set of two-hop neighbors $a_1, a_2...., a_n$ sorted with respect to their addresses. Under the condition of valid change of codes, no node can have the same code as a two-hop neighbor with lesser address, i.e a_2 cannot have the same code as a_1, a_3 cannot have the same code as a_1 and a_2. If the condition of valid change of codes is true for all nodes, any node a_i cannot have the same code as $a_{i-1}, a_{i-2}, ..., a_1$. If we continue this argument till a_n, we see that a_n cannot have the same code as $a_{n-1}, a_{n-2}...a_1$. From this we see that for a set of two-hop neighbors with a bounded number of nodes, no two nodes have the same code. One of the assumptions we made is that we have a finite network, i.e. a bounded number of nodes. Therefore, we have a correct code assignment in which no two-hop neighbors have the same code.

IV. Complexity of the Code Assignment Algorithm

This section gives results about the communication complexity(i.e. the number of messages needed in the worst case), computation complexity and storage complexity after a single code change.

A. Communication Complexity

There can be two results of a node changing its code.

1. The node's new code is not the same as any of its two-hop neighbors' codes. In this case, there are only $O(d^2)$ messages exchanged.
2. The node's new code is the same as one of its two-hop neighbors' code. In this case, the two-hop neighbor might have to change code which in turn could cause CAM's to be sent. In a pathological case, a node's change of code might cause all the nodes in the network to change their codes.

From the above, it follows that the number of messages is bounded by $O(|V|.d^2)$.

B. Complexity of Computation at the Nodes

In the worst case there are $d+1$ entries in the CAM. This includes new entries which did not exist in the priority list earlier. The entries in the CAM are presumed to be in the sorted order. A scan of the priority list is done and the new codes added to it. While doing the scan we can check if any of the two-hop neighbors have the same code and also update the UCL. There are $d(d-1)+d+1$ entries in the priority list. A linear scan will take the order of $d(d-1)+d+1$ which is $O(d^2)$.

C. Storage Complexity

The priority list has $d(d-1)+d+1$ entries. The MRL can have a specified maximum number of entries which is presumed to be a constant. The UCL has to have a minimum of $d(d-1)+2$ entries. Thus the storage complexity is $O(d^2)$.

V. Conclusions

We have presented an algorithm for distributed, dynamic channel assignment in multihop packet radio networks. This protocol is based on obtaining information about the transmission codes used by nodes one-hop and two-hops away. The algorithm was shown to be correct and its complexity was analyzed. This protocol fits in nicely with existing MAC and routing protocols designed for multihop packet-radio networks. At UCSC, we are implementing the proposed algorithm as part of the FAMA [8] and Wireless Internet Routing Protocol [12] which are used in the Wireless Internet Gateways [2], [7].

References

[1] Ricochet Wireless Network. *http://www.ricochet.net/netoverview.html*.

[2] Wireless Internet Gateways. *http://www.cse.ucsc.edu/research/ccrg /projects/wings.html*.

[3] A.A. Bertossi and M.A. Bonuccelli. Code Assignment for Hidden Terminal Interference Avoidance in Multihop Radio Networks. *IEEE/ACM Trans. on Networking*, 3(4):441–449, August 1995.

[4] I. Chlamtac and S. Kutten. Tree-Based Broadcasting in Multihop Radio Networks. *IEEE Trans. Computers*, Oct 1987.

[5] I. Cidon and M. Sidi. Distributed Assignment algorithms for Multihop Packet-Radio Networks. *Proc. IEEE Infocom*, pages 1110–1118, 1988.

[6] A. Ephremides and T.V. Truong. Scheduling Broadcasts in Multihop Radio Networks. *IEEE Trans. Computers*, 38(4):456–460, April 1990.

[7] J.J. Garcia-Luna-Aceves et. al. Wireless Internet Gateways (WINGS). In *Proc. IEEE MILCOM 97,*, Monterey, CA, 1997.

[8] C.L. Fullmer and J.J. Garcia-Luna-Aceves. FAMA-PJ:A Channel Access Protocol for Wireless LANs. *Proc. ACM Mobile Computing and Networking*, 1995.

[9] L. Hu. Distributed Code Assignment for CDMA Packet Radio Networks. *IEEE/ACM Trans. on Networking*, 1(6):668–677, Dec 1993.

[10] IEEE. *P802.11-Unapproved Draft. Wireless LAN Medium Access Control(MAC) and Physical Specifications*, Nov 1991.

[11] T. Makansi. Trasmitter-Oriented Code Assignment for Multihop Radio Networks. *IEEE Trans. Computers*, COM-35(12):1379–1382, Dec 1987.

[12] S. Murthy and J.J. Garcia-Luna-Aceves. An Efficient Routing Protocol for Wireless Networks. *ACM Mobile Networks and Applications Journal*, 1(2):183–197, Oct 1996.